Desafíos y herramientas para la enseñanza temprana de Concurrencia y Paralelismo

Laura De Giusti[1], Fabiana Leibovich[1], Mariano Sánchez[1], Franco Chichizola[1], Marcelo Naiouf[1], Armando De Giusti[1,2]

[1] Instituto de Investigación en Informática LIDI (III-LIDI) – Facultad de Informática –UNLP
[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Argentina
{ldgiusti, fleibovich, msanchez, francoch, mnaiouf, degiusti}@lidi.info.unlp.edu.ar

Abstract. Se analiza la introducción temprana de los temas básicos de concurrencia y paralelismo, de acuerdo a las tendencias curriculares impulsadas por el cambio tecnológico.
El artículo analiza la problemática a partir de la introducción de los procesadores de múltiples núcleos y las nuevas arquitecturas asociadas con Cluster, Multicluster y Clouds basados en arquitecturas multicore / GPGPU.
Se presenta una herramienta que combina un entorno visual interactivo para la programación concurrente, con el empleo de robots de demostración, especialmente para los temas de comunicación y sincronización.
Por último se analizan aplicaciones que extienden el alcance del entorno desarrollado, su aplicación en diferentes cursos y las líneas de I/D futuras en el tema.

Keywords: Concurrencia, Paralelismo, Currícula, Entorno, Multirobot, Algoritmos Concurrentes y Paralelos.

1 Introducción

La Concurrencia ha sido un tema central en el desarrollo de la Informática y los mecanismos de expresión de procesos concurrentes que cooperan y compiten por recursos ha estado en el núcleo curricular de los estudios de Informática desde la década del 70, en particular a partir de los trabajos fundacionales de Hoare, Dijkstra y Hansen [HOA78][HOA85][DIJ65][DIJ78][HAN77].
Estos conceptos se enseñaron tradicionalmente partiendo de la disponibilidad de un único procesador central, que podía explotar parcialmente la concurrencia de un dado algoritmo, en función de la arquitectura física disponible (incluso con hardware específico como los coprocesadores, los controladores de periféricos o esquemas vectoriales que replicaban las unidades de cómputo aritmético-lógico).
El paralelismo, entendido como "concurrencia real" en la que múltiples procesadores pueden operar simultáneamente sobre múltiples threads o hilos de control en el mismo instante de tiempo, resultó durante muchos años una posibilidad limitada por la tecnología de hardware disponible [HWA84][HWA93][DAS89].

En las currículas informáticas clásicas [ACM68][ACM78][ACM99] aparecían los conceptos de concurrencia en diferentes áreas (Lenguajes, Paradigmas, Sistemas Operativos) y se omitía casi totalmente el tratamiento del paralelismo, salvo al plantear los conceptos de sistemas distribuidos.
La aparición del lenguaje ADA [OLS83] a mediados de los 80 marca un hito en la evolución del tema, ya que especifica claramente en un lenguaje real los diferentes mecanismos de expresión de la concurrencia y al mismo tiempo deja clara la posibilidad de asociar los procesos ("tasks" en ADA) a diferentes procesadores físicos.
Las nuevas arquitecturas de los procesadores, que integran múltiples "cores" o núcleos en un procesador físico [GEP06][MCC08][GPG] han producido un notorio impacto en el desarrollo de la Informática, obligando a replantear el "modelo base" de un procesador. Esto ha llevado a reemplazar el formato de "máquina de Von Neuman" [GOL72] con un solo hilo de control, por un esquema como el de la Figura 1 que integra múltiples "cores" cada uno con uno o más hilos de control y varios niveles de memoria accesible en forma diferenciada [AMD09].

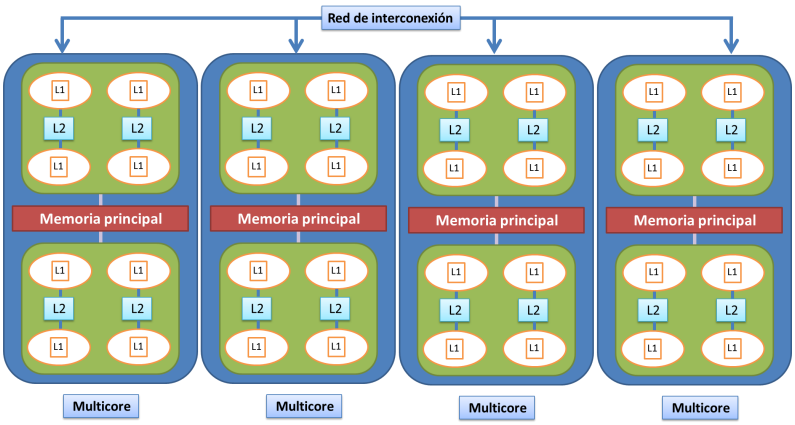

Fig. 1 Esquema de un procesador básico actual.

Al mismo tiempo, los cambios tecnológicos han producido una evolución de los temas de mayor interés en Informática, fundamentalmente por las nuevas aplicaciones que se desarrollan a partir de disponer de arquitecturas y redes de comunicación de mayor potencia y menor costo [DEG13][HOO13].
Esto ha llevado a que las recomendaciones curriculares internacionales [ACM04] [ACM08][ACM13] mencionen la necesidad de tratar los temas de concurrencia y paralelismo en etapas tempranas de la formación del alumno, dado que todas las arquitecturas y sistemas reales con los que trabajará son esencialmente paralelos.
Sin embargo, aquí aparece uno de los problemas importantes, ya que la programación paralela (y los conceptos fundamentales de concurrencia) resulta más compleja para un alumno en las etapas iniciales de su formación. Es necesario contar con nuevas herramientas que permitan abordar tempranamente el tema [CAR03][DEG12a].

2 Objetivos del entorno multirrobot en desarrollo

En este trabajo se presenta el entorno CMRE (Concurrent Multi Robot Environment) como una herramienta destinada a introducir los conceptos de concurrencia y paralelismo, con un enfoque visual e interactivo combinado con el empleo de robots físicos para la demostración de los conceptos y ejemplos de desarrollo.

2.1 Entorno visual en Concurrencia y Paralelismo

Se ha partido del trabajo desarrollado en el III-LIDI [DEG12b][DEG12c] para la enseñanza de conceptos de concurrencia en un curso CS1 buscando extender el mismo en los siguientes ejes:

- Poder declarar "procesadores" o robots virtuales que representan los "cores" de una arquitectura multiprocesador real. Estos robots virtuales pueden tener un reloj propio y diferentes tiempos para la ejecución de sus tareas específicas.
- Tener la capacidad de declarar recursos compartidos o exclusivos, incluyendo la posibilidad de tener exclusión mutua selectiva.
- Establecer objetos virtuales que representen datos básicos que se pueden contar y manipular en forma simple mediante primitivas de los robots virtuales.
- Disponer de primitivas de comunicación por mensajes sincrónicos y/o asincrónicos.
- Disponer de primitivas de sincronización por memoria compartida.

2.2 Incorporación de robots físicos al entorno visual

A los puntos anteriores se le incorpora la comunicación en tiempo real con robots físicos de la línea Lego Mindstorms EV3 [LEGa][LEGb] de modo de poder ejecutar algoritmos paralelos en el entorno, con efecto directo en los robots físicos que replican sobre el terreno el comportamiento definido por los algoritmos.
Este modelo de demostración facilita la comprensión de determinados problemas por parte del alumno, tales como los conceptos de *fairness, deadlock o inanición* [AND00].

3 Arquitectura y Primitivas de Concurrencia/Paralelismo en CMRE. Ejemplos

El entorno CMRE surge como una evolución del entorno Visual da Vinci cuyo objetivo principal fue resolver problemas donde se especifica el comportamiento de un *único robot*, el cual puede moverse en una ciudad compuesta por 100 avenidas (verticales) y 100 calles (horizontales) y es capaz de distinguir objetos (flores y papeles) y realizar operaciones con los mismos (juntarlos y/o depositarlos). Asimismo el robot puede "contar" e "informar" resultados. En la Tabla 1 se sintetiza la metáfora buscada con el nuevo entorno.

Tabla 1. Analogía entre el entorno CMRE y los conceptos de Concurrencia y Paralelismo

Conceptos de Concurrencia y Paralelismo	Entorno CMRE
Múltiples procesadores / cores	Múltiples robots (implementado con un proceso por robot)
Memoria compartida	Áreas compartidas de la ciudad
Memoria distribuida	Áreas exclusivas por robot
Memoria compartida y distribuida	Áreas parcialmente compartidas
Comunicación entre procesos por mensajes	Envío y recepción de mensajes entre robots.
Exclusión mutua sobre recursos compartidos	Bloqueo de esquinas de la ciudad.
Exclusión mutua selectiva	Acceso a áreas parcialmente compartidas.
Modelo de ejecución sincrónico	Reloj virtual sincrónico.
Arquitecturas heterogéneas	Asignar tiempos específicos a las operaciones de cada robot.
Datos locales o globales	Objetos numerables en la ciudad (flores/papeles).

En la Figura 2, a modo ilustrativo se define la estructura general de un programa en el entorno CMRE, en función del cual se especificarán sus primitivas.

```
programa areas1
areas      {defino la estructura de la ciudad}
  nombreArea1: tipoArea(Coordenada0, Coordenada1, Coordenada2, Coordenada3)
  nombreArea2: tipoArea(Coordenada0, Coordenada1, Coordenada2, Coordenada3)
robots       {defino el comportamiento de cada tipo de robot}
  robot tipo1
  comenzar
    {cuerpo}
  fin
  robot tipo2
  comenzar
    {cuerpo}
  fin
variables      {creo los robots}
   nombreVariableRobot1: tipo1
   nombreVariableRobot2: tipo2
comenzar
  {Asigno areas privadas a cada robot}
  AsignarArea(nombreVariableRobot1, nombreArea1)
  AsignarArea(nombreVariableRobot2, nombreArea2)
  iniciar(nombreVariableRobot1, PosAv, PosCa)
  iniciar(nombreVariableRobot1, PosAv, PosCa)
fin
```

Fig. 2. Estructura general de un programa en el entorno CMRE.

Tal como se mencionó anteriormente pueden resumirs las capacidades del ambiente CMRE de la siguiente manera:

- Existen múltiples procesadores (robots) que realizan tareas y que pueden cooperar y/o competir.
- El modelo de ambiente ("ciudad") en la que desarrollan sus tareas admite áreas privadas, parcialmente compartidas y totalmente compartidas. En un área privada sólo puede moverse un único robot, en un área parcialmente compartida se especifica el conjunto de robots que pueden moverse en ella y en un área totalmente compartida todos los robots definidos en el programa pueden moverse dentro de ella.
- Si se instancia un sólo robot en un área que abarque toda la ciudad, se repite el esquema del Visual Da Vinci.
- Cuando dos o más robots están en un área compartida (parcial o totalmente), compiten por el acceso a las esquinas del recorrido y a los recursos que allí existan. Para esto deben sincronizar.
- Cuando dos o más robots (en un área común o no) desean intercambiar información (datos o control) deben hacerlo por mensajes explícitos.
- La sincronización se da por un mecanismo equivalente a un semáforo binario.
- La exclusión mutua puede generarse con la declaración de las áreas alcanzadas por cada robot. Acceder a otras áreas de la ciudad, así como salir de ellas no está permitido.
- Todo el modelo de ejecución es sincrónico y permite la existencia de un reloj virtual de ciclos, que a su vez permite asignar tiempos específicos a las operaciones, simulando la existencia de una arquitectura heterogénea.
- El entorno permite ejecutar el programa de manera tradicional, o paso a paso por instrucciones, dando al usuario un control detallado sobre la ejecución del programa, de manera de poder controlar situaciones típicas de concurrencia tales como conflictos (colisiones) o deadlocks.
- En la ejecución paso a paso, el efecto de las operaciones se puede reflejar en los robots físicos, comunicados vía WI-FI.
- En el entorno, cada robot tiene asociado un estado, en el que muestra el contenido de su bolsa (cantidad de flores y papeles en el modelo), esquina donde se encuentra situado, estado actual: si se encuentra ejecutando, esperando la llegada de un mensaje, o esperando por la liberación de una esquina.

3.1 Declaración de áreas

La declaración de áreas comienza con la palabra clave **areas** y termina donde se encuentra la palabra clave **robots**. Un área de la ciudad es un subconjunto rectangular de esquinas de la ciudad por la que los robots pueden circular. Estas pueden ser de tres tipos:

- Área compartida (areaC): es el tipo de área por defecto y corresponde a una región de la ciudad de libre acceso, es decir, cualquier robot puede circular por ella.
- Área privada (areaP): una región de este tipo sólo permite que haya un robot en ella. El intento de un robot de ingresar en un área privada de otro, genera un error en tiempo de ejecución. Notar que las áreas privadas permiten un mecanismo de

exclusión mutua implícita entre robots.

- Área parcialmente compartida (areaPC): este tipo de regiones permiten el acceso de uno o varios robots, con la restricción de que deben haber sido autorizados previamente. Notar que las áreas parcialmente compartidas permiten un mecanismo de exclusión mutua selectiva entre robots.

Cada declaración de área comienza con un nombre, seguido de dos puntos y la palabra clave areaC, areaP o areaPC (para indicar su tipo) más cuatro parámetros. Éstos representan las coordenadas inferior izquierda y superior derecha que ocupará el área dentro de la ciudad. Cada tipo de área tiene asociado un color, tal como muestra en la Figura 3.

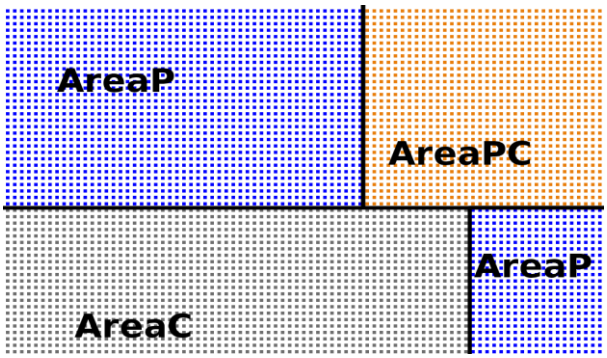

Fig. 3. Esquema de la ciudad con las definiciones de las distintas áreas.

El alcance y la visibilidad de las áreas corresponden a todo el programa, y deben ser asignadas a los robots sobre los que se quiera permitir su acceso antes de comenzar la ejecución de los mismos.

3.2 Declaración de robots

La declaración de robots comienza con la palabra clave **robots** y termina donde se encuentra la palabra clave **comenzar**.

Los robots tienen una estructura casi idéntica a la del programa principal o subprocesos, incluyendo encabezado, declaraciones y cuerpo.

El encabezado comienza con la palabra clave **robot** seguido de un nombre. Las declaraciones de procesos y variables locales siguen las mismas reglas que las declaraciones del programa principal, con la salvedad que no pueden declararse nuevas áreas o robots.

De esta manera, la creación de robots siempre es explicita. Una característica a destacar es que aun cuando se quiera utilizar el entorno para simular un ambiente como el del Visual Da Vinci, será necesario crear el único robot desde el programa principal.

El cuerpo de un robot es una secuencia de sentencias, delimitada por las palabras clave **comenzar** y **fin.** Estas sentencias se corresponden con las de Visual Da Vinci, y sirven para definir el comportamiento del robot en la ciudad.

La posibilidad de definir tipos de robots permite reutilizar el código para diferentes

robots que llevan a cabo el mismo comportamiento, teniendo en cuenta que se debe tener cuidado principalmente con el uso de ubicaciones absolutas, dado que los robots puede que compartan o no un área de la ciudad.

3.3 Cuerpo del programa principal

Desde el programa principal es necesario asignar los robots a la áreas que pertenecen y luego "arrancar" cada uno ellos mediante la directiva *Iniciar*, que requiere del nombre del robot y su ubicación inicial.
Esta sentencia requiere de comprobaciones en tiempos de compilación para que dos o más robots no intenten ocupar la misma esquina, teniendo en cuenta también a qué tipo de área pertenece la esquina involucrada. Para dar un ejemplo, un robot no puede ocupar una esquina que pertenece a un área exclusiva de otro robot.
A estas sentencias se agrega un nuevo subconjunto, denominado sentencias de concurrencia.

3.4 Manejo de colisiones

Conceptualmente los "recursos compartidos" en este modelo de entorno se pueden reducir al acceso a una esquina, donde puede haber objetos.
Evitar las "colisiones" en las esquinas es el problema básico de sincronización en el entorno CMRE.
Para esto el lenguaje cuenta con directivas que permiten bloquear y liberar el recurso:

- *bloquearEsquina* (BE): indica que el robot pide exclusión para la ocupación de una esquina (lo que a su vez le permite recoger o depositar objetos).
- *liberarEsquina* (LE): indica que el robot deja el recurso libre (la esquina ocupada).

3.5 Comunicación / Sincronización

Tal como se mencionó anteriormente, hay múltiples robots que trabajan en la ciudad. En muchos casos, deberán colaborar en la resolución de algún problema.
Esto requiere comunicación y sincronización. Se adopta un mecanismo explícito de pasaje de mensajes asincrónicos con dos directivas:

- *enviarMensaje* (EM): permite que un robot envíe un mensaje a otro (identificados por su nombre). Al enviar el mensaje, según el modelo asincrónico, el robot continúa con la siguiente instrucción secuencialmente sin esperar la recepción.
- *recibirMensaje* (RM): indica que un robot se quedará esperando hasta sincronizar con el envío de mensaje de otro. En la recepción se indica el nombre del robot del cual se espera el mensaje.

Para finalizar se eligieron dos problemas ampliamente utilizados en la enseñanza de los conceptos de programación concurrente y paralela.
En la Figura 4 se muestra el código correspondiente a un problema "master/worker" que utiliza Pasaje de Mensajes. Para esto se declaran 4 aéreas privadas junto a 4 robots (1,2,3 workers ,4 master) donde cada uno interactúa en su área privada juntado todas las flores que existen en ella, y al finalizar los robots 1, 2 y 3 envían sus resultados al 4 para que este los totalice e informe dicho total.

```
programa ejemplo1

procesos
  proceso avenida(ES f:numero)
  comenzar
    repetir 49
      {recorre la avenida y junta las flores
        acumulando en el parámetro f}
  fin

areas
  {definición de áreas}
  area1: areaP(1, 1, 50, 50)
  area2: areaP(1, 51, 50, 100)
  area3: areaP(51, 1, 100, 50)
  area4: areaP(51, 51, 100, 100)

robots
  {comportamiento de c/tipo de robot}
  robot worker
  variables
    f:numero
  comenzar
    f:=0
    repetir 49
      avenida(f)
      Pos(PosAv+1,PosCa-49)
    avenida(f)
    enviarMensaje(f, robot4)
  fin
```

```
robot master
  variables
    f:numero
    total:numero
  comenzar
    f:=0
    repetir 49
      avenida(f)
      Pos(PosAv+1,PosCa-49)
    avenida(f)
    total:=f
    recibirMensaje(f, robot1)
    total:=total+f
    recibirMensaje(f, robot2)
    total:=total+f
    recibirMensaje(f, robot3)
    total:=total+f
    informar(total)
  fin

variables
  {creación variables robots}
  robot1: worker
  robot2: worker
  robot3: worker
  robot4: master

comenzar
  {Asignación de áreas a cada robot}
  AsignarArea(robot1, area1)
  AsignarArea(robot2, area2)
  AsignarArea(robot3, area3)
  AsignarArea(robot4, area4)
  iniciar(robot1, 1, 1)
  iniciar(robot2, 1, 51)
  iniciar(robot1, 51, 1)
  iniciar(robot2, 51, 51)
fin
```

Fig. 4. Ejemplo de programa con Pasaje de Mensajes.

En la Figura 5 se muestra el código correspondiente a un problema que utiliza Memoria Compartida. Para esto se declara que toda la ciudad es compartida por 2 robots (1 y 2) donde deben coordinarse para trasladar de a una las flores de la esquina (1,1) hasta que la misma queda vacía. Esta coordinación debe darse para garantizar que los robots no estén en la misma esquina simultáneamente, y por consiguiente no tomen la misma flor. Cada vez que un proceso toma una flor la traslada a la esquina siguiente (diferente para cada robot).

```
programa ejemplo2
procesos
  proceso girar(E cant:numero)
  comenzar
    repetir cant
      derecha
  fin
  proceso depositarUnaFlor
  comenzar
    mover
    liberarEsquina(1,1)
    depositarFlor
  fin

areas
  {definición de áreas}
  area1: areaC(100, 100, 100, 100)

robots
  {comportamiento de c/tipo de robot}
  robot tipo1
  variables
    seguir:boolean
  comenzar
   seguir:=V
    girar(2)
    bloquearEsquina(1,1)
    mover
    si ~(HayFlorEnLaEsquina)
       seguir:=F
    mientras(seguir)
      tomarFlor
      girar(2)
      depositarUnaFlor
      girar(2)
      bloquearEsquina(1,1)
      mover
      si ~(HayFlorEnLaEsquina)
         seguir:=F
    girar(2)
    mover
    liberarEsquina(1,1)
  fin
```

```
  robot tipo2
  variables
    seguir:boolean
  comenzar
    seguir:=V
    girar(3)
    bloquearEsquina(1,1)
    mover
    si ~(HayFlorEnLaEsquina)
      seguir:=F
    mientras(seguir)
      tomarFlor
      girar(2)
      depositarUnaFlor
      girar(2)
      bloquearEsquina(1,1)
      mover
      si ~(HayFlorEnLaEsquina)
        seguir:=F
    girar(2)
    mover
    liberarEsquina(1,1)
  fin

variables
  {creación variables robots}
  robot1: tipo1
  robot2: tipo2

comenzar
  {Asignación de áreas a cada robot}
  AsignarArea(robot1, area1)
  AsignarArea(robot2, area1)
  iniciar(robot1, 1, 2)
  iniciar(robot2, 2, 1)
fin
```

Fig. 5. Ejemplo de programa en Memoria Compartida.

3.6 Estado del desarrollo actual

El entorno CMRE está totalmente desarrollado en Java y se está utilizando experimentalmente en la UNLP. Los robots físicos están en proceso de compra, aunque no introducen (en el estado actual del desarrollo) una complejidad adicional.

Las características elegidas de los robots físicos se relacionan con nuevas posibilidades que se abren para el entorno, tal como se indica en las líneas de trabajo futuras.

4 Conclusiones y Líneas de Trabajo Futuro

Se ha presentado un entorno para la enseñanza temprana de los conceptos de concurrencia y paralelismo, asociados con el empleo de robots virtuales y físicos en un entorno de programación interactivo y flexible.
Actualmente se está estudiando la generalización del empleo del CMRE en aplicaciones en las cuales los robots adquieren información en tiempo real y los algoritmos definidos en el entorno toman decisiones dinámicamente. Esto es de particular importancia para asignaturas relacionadas con Sistemas de Tiempo Real e incluso con Sistemas Inteligentes.

5 Bibliografía

[ACM04] ACM/IEEE-CS Joint Task Force on Computing Curricula. "Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering". Report in the Computing Curricula Series. 2004.

[ACM08] ACM/IEEE-CS Joint Interim Review Task Force. "Computer Science Curriculum 2008: An Interim Revision of CS 2001". Report from the Interim Review Task Force. 2008.

[ACM13] ACM/IEEE-CS Joint Task Force on Computing Curricula. "Computer Science Curricula 2013". Report from the Task Force. 2013.

[ACM68] ACM Curriculum Committee on Computer Science. "Curriculum „68: Recommendations for the undergraduate program in computer science". Communications of the ACM, 11(3):151-197. 1968.

[ACM78] ACM Curriculum Committee on Computer Science. "Curriculum „78: Recommendations for the undergraduate program in computer science". Communications of the ACM, 22(3):147-166. 1979.

[ACM99] ACM Two-Year College Education Committee. "Guidelines for associate-degree and certificate programs to support computing in a networked environment". New York: The Association for Computing Machinery. 1999.

[AMD09] AMD. "Evolución de la tecnología de múltiple núcleo". http://multicore.amd.com/es-ES/AMD-Multi-Core/resources/Technology-Evolution. 2009.

[AND00] Andrews G. "Foundations of Multithreaded, Parallel, and Distributed Programming". Addison Wrsley, 2000.

[CAR03] Carr S., Mayo J., Shene C. "Threadmentor: a pedagogical tool for multithreaded programming". ACM Journal of Educational Resources, 3:1–30, 2003.

[DAS89] Dasgupta S. "Computer Architecture. A Moder Synthesis. Volume 2: Advanced Topics". Jhon Wilet & Sons. 1989.

[DEG12a] De Giusti A. E., Frati F. E., Leibovich F., Sánchez M., De Giusti L. C., Madoz M. C. "Concurrencia y Paralelismo en CS1: la utilización de un Lenguaje Visual orientado". Proceeding del VII Congreso de Tecnología en Educación y Educación en Tecnología. 2012

[DEG12b] De Giusti L. C., Frati F. E., Leibovich F., Sánchez M., Madoz M. C. "LMRE: Un entorno multiprocesador para la enseñanza de conceptos de concurrencia en un curso CS1".

Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología. Págs. 7 - 15. 2012.
[DEG12c] De Giusti A. E., Frati F. E., Sánchez M., De Giusti L. C. “LIDI Multi Robot Environment: Support software for concurrency learning in CS1”. Proceeding of IEEE International Conference on Collaboration Technologies and Systems. Pág. 294-298. 2012.
[DEG13] De Giusti A. E. “El cambio tecnológico como motor de la Investigación en Informática”. Conferencia inaugural del Workshop de Investigadores en Ciencia de la Computación (WICC2013). 2013.
[DIJ65] Dijkstra E. W. “Solution of a problem in concurrent programming control”. Communications of the ACM, 8(9):569, 1965.
[DIJ78] Dijkstra E. W. “Finding the Correctness Proof of a Concurrent Program”. In Program Construction, International Summer Schoo, Friedrich L. Bauer and Manfred Broy (Eds.). Springer-Verlag, 24-34, 1978.
[GEP06] Gepner P., Kowalik M.F. “Multi-Core Processors: New Way to Achieve High System Performance”. In: Proceeding of International Symposium on Parallel Computing in Electrical Engineering 2006 (PAR ELEC 2006). Pags. 9-13. 2006.
[GOL72] Goldstine H. H. “The Computer”. Princeton University Press, 1972.
[GPG] GPGPU. “General-Purpose Computation on Graphics Processing Units”. http://gpgpu.org.
[HAN77] Hansen P. B. “The Architecture of Concurrent Processes”. Prentice Hall, 1977.
[HOA78] Hoare C. “Communicating Sequential Processes”. Communications of the ACM, 21(8): 666-677, 1978.
[HOA85] Hoare C. “Communicating Sequential Processes”. Prentice Hall, 1985.
[HOO13] Hoonlor A., Szymanski B. K., Zaki M. J., Thompson J. “An Evolution of Computer Science Research”. Communications of the ACM. 2013.
[HWA84] Hwang K., Briggs F. A. “Computer Architecture and Parallel Processing”. McGraw Hill, 1984.
[HWA93] Hwang K. “Advanced Computer Architecture: Parallelism, Scalability, Programmability”. McGraw Hill, 1993.
[LEGO] "Lego Education". http://www.legoeducation.us/eng/characteristics/ProductLine~LEGO%20MINDSTORMS%20Education%20EV3.
[LEGOb]Lego. "LEGO Mindstorms EV3 Announced". http://brickextra.com/2013/01/10/lego-mindstorms-ev3-announced/
[MCC08] McCool M. “Scalable Programming Models for Massively Parallel Multicores”. Proceedings of the IEEE, 96(5): 816–831, 2008.
[OLS83] Olsen E. W., Whitehill S. B. “Ada for Programmers”. Prentice Hall, 1983.